

MULTIOPERATING SYSTEM CONTROL METHOD

Publication number: JP2000076087 (A)

Publication date: 2000-03-14

Inventor(s): ONO HIROSHI; NAKAMURA TOMOAKI; KANEKO SHIGENORI; YOSHIZAWA RYOKICHI; KATO SUNAO; YAMAUCHI MANABU; ARAI TOSHIKI; SEKIGUCHI TOMONORI

Applicant(s): HITACHI LTD

Classification:

- international: G06F9/50; G06F9/46; G06F9/46; (IPC1-7): G06F9/46; G06F9/46

- European:

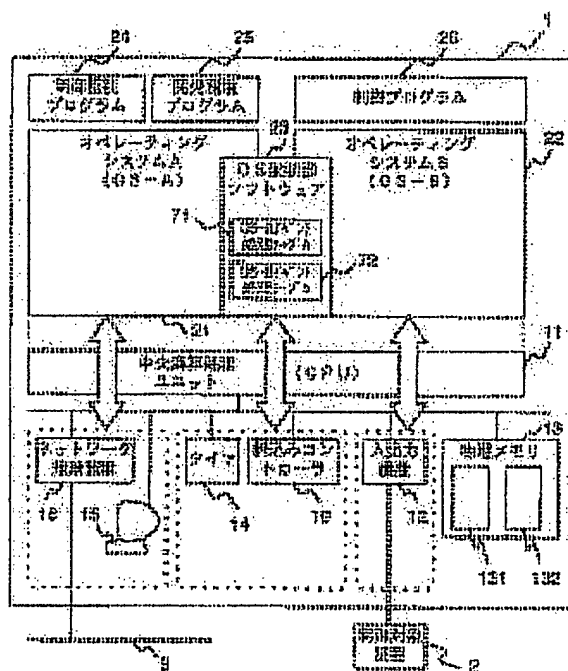
Application number: JP19980242833 19980828

Priority number(s): JP19980242833 19980828

Abstract of JP 2000076087 (A)

PROBLEM TO BE SOLVED: To localize the influence of abnormality and to improve the reliability when more than one OS is run by mounting inter-OS control software which switches an OS in operation so that OSs operate alternately on the same CPU.

SOLUTION: The software of a controller 1 consists of inter-OS control software 23, a control and monitor program 24 and a development environment program 25 which run on an operating system OS-A21 and a control program 26 which runs on an operating system OS-B22 in addition to the A(OS-A) 21 and B(OS-B)22 which perform hardware resource management and the execution management of programs running above. Exclusively managed hardware is assigned to those OS-A21 and OS-B22 and the execution right of the OSs is switched alternately; and the OSs operate independently after the switching. The inter-OS control software 23 switches the operations of both the OSs and provides a communication function between both OSs.



(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2000-76087

(P2000-76087A)

(43) 公開日 平成12年3月14日 (2000.3.14)

(51) Int.Cl. ⁷	識別記号	F I	テマコード* (参考)
G 0 6 F 9/46	3 5 0	G 0 6 F 9/46	5 B 0 9 8
	3 4 0		3 4 0 A

審査請求 未請求 請求項の数 3 O L (全 10 頁)

(21) 出願番号 特願平10-242833

(22) 出願日 平成10年8月28日 (1998.8.28)

(71) 出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(72) 発明者 大野 洋

茨城県日立市大みか町五丁目2番1号 株

式会社日立製作所大みか工場内

(72) 発明者 中村 智明

茨城県日立市大みか町五丁目2番1号 株

式会社日立製作所大みか工場内

(74) 代理人 100068504

弁理士 小川 勝男

最終頁に続く

(54) 【発明の名称】 マルチオペレーティングシステム制御方法

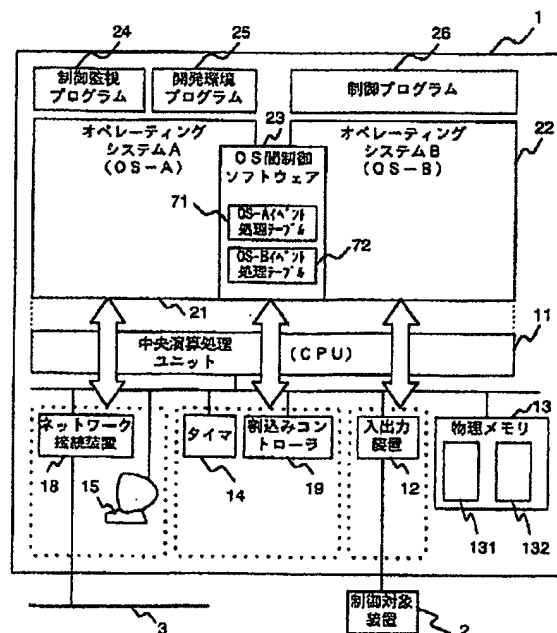
(57) 【要約】

【課題】 単一のCPU上で複数のOSを動作させて制御システムを構成する場合、制御プログラム以外のプログラムおよびデバイスドライバの動作が、制御プログラムの動作に影響を与える。

【解決手段】 単一のCPU上で、動作中のOSを切替えるOS間制御ソフトウェアを搭載し、複数のOSが交互に動作するようにしたものである。一方のOS上では、制御プログラムを専用で動作させ、これにより制御対象装置を制御する。もう一方のOS上では制御監視プログラムおよび開発環境プログラムが動作するが、制御プログラムの動作には影響を与えないようにメモリ空間を分離する。

【効果】 単一のCPUで、高い実時間処理性能と信頼性を達成可能である。

図 1



【特許請求の範囲】

【請求項1】 デジタル演算プロセッサにて動作する複数のオペレーティングシステムを搭載し、当該複数のオペレーティングシステムの内、第1のオペレーティングシステムと第2のオペレーティングシステムとが前記デジタル演算プロセッサにて交互に動作を切替えるマルチオペレーティングシステム制御方法において、

前記第1のオペレーティングシステムの起動時に、前記第2のオペレーティングシステムが使用する割り込み番号あるいは入出力アドレスを前記第1のオペレーティングシステムに対して予約する処理を備えたことを特徴とするマルチオペレーティングシステム制御方法。

【請求項2】 請求項1に記載のマルチオペレーティングシステム制御方法において、前記デジタル演算プロセッサが仮想メモリアドレスから物理アドレスに変換するために使用する変換テーブルが、前記各オペレーティングシステム毎に存在し、

割り込み発生時に割り込み処理ルーチンとして呼出され又は前記第1及び第2の各オペレーティングシステムから呼出され、前記各オペレーティングシステムのうちいずれか一つを選択して切替える切替え処理の際に、オペレーティングシステムを選択して切替える時点で、切替え後に動作させるオペレーティングシステムの前記変換テーブルを使用するように前記デジタル演算プロセッサに対して指示する処理を行うことを特徴とするマルチオペレーティングシステム制御方法。

【請求項3】 請求項1に記載のマルチオペレーティングシステム制御方法において、

動作している前記各オペレーティングシステムに優先順位を付け、

優先順位の高いオペレーティングシステム上で動作するプログラムの処理がある間は、より優先順位の低いオペレーティングシステムへの動作切替えを行わないか、または一定の時間割合のみ動作を切替えその後高優先順位のオペレーティングシステムへ動作を戻すことにより、前記優先順位の高いオペレーティングシステム上で動作するプログラムを優先して実行することを特徴とするマルチオペレーティングシステム制御方法。

【請求項4】 請求項1に記載のマルチオペレーティングシステム制御方法において、

動作している前記各オペレーティングシステムに優先順位を付け、優先順位の高いオペレーティングシステム上で動作するプログラムの処理がある間は、より優先順位の低いオペレーティングシステムへの動作切替えを行わないか、または一定の時間割合のみ動作を切替えその後高優先順位のオペレーティングシステムへ動作を戻すことにより、前記優先順位の高いオペレーティングシステム上で動作するプログラムを優先して実行することを特徴とするマルチオペレーティングシステム制御方法。

らの制御装置に対して制御論理を入力する機能（開発環境）および、制御結果の監視・表示あるいは対話型でデータを入力する機能（ヒューマン・マシン・インタフェース）などの、制御論理処理を除く機能については、制御装置の外部にパーソナルコンピュータ（PC）などの別な装置を接続して実現することが多い（以下、これを「ユーザインタフェース装置」と呼ぶ）。またこれらの機能が同一装置内に具備されている場合でも、内部で異なる演算プロセッサを使用して機能分散して実現している。この種の装置として関連するものには例えば特開平9-62324号公報に記載の技術が挙げられる。

【0003】 一方、ユーザインタフェース装置として使用されてきたPCの性能が向上し、ある程度の規模までならば、制御論理処理から開発環境、制御監視の機能までをすべて1台のPCで実行することが可能な演算能力が提供されるようになってきた。しかしPCを基本としたハードウェアにすべての機能を含んだ制御装置の場合、PCで広く用いられているオペレーティングシステム（OS）を用いた場合、制御プログラム以外のプログラムまたはデバイスドライバの動作が、制御プログラムの動作に影響を与える可能性がある。

【0004】 そこで、同一CPU上で複数のOSを同時に起動し、計算機資源の共有を図る一方で個々のOSが持つ機能を利用する技術がある。このような例としては、特開平5-73340号公報および特開平5-27954号公報、特開平5-151003号公報等に記載の技術がある。

【0005】

【発明が解決しようとする課題】 通常OSでは特権命令を実行する。その為、あるOS自体に障害が発生すると、他のOSの実行に影響を与えてしまう。しかし、上記の1台の計算機上で複数のOSを同時に起動する技術においては、この点に対する考慮がされておらず、前述の特開平5-151003号のように、一方のOSを他方のOS上でエミュレートすることにより、一方のOSにおける障害の影響を切離しても、エミュレートする側のOSに障害が発生した場合には、両方のOSの動作に影響を与えることになる。

【0006】

【課題を解決するための手段】 本発明では上記課題を解決するために、同一CPU上で、複数のOSが交互に動作するように動作中のOSを切替えるOS間制御ソフトウェアを搭載する。そして、制御プログラムを信頼性の高いOSで、ユーザインタフェースを機能の豊富なOSでというように、特性の異なるOSを同一のCPU上で実行させる。このOS間制御ソフトウェアは、割り込みの発生、あるいはOSまたはその上で動くソフトウェアからの要求を契機として、動作中のコンテキスト情報（CPUのレジスタ値等）を保存し、メモリ空間の切替えを行い、以前に保存されていた別なコンテキストで実行を再開する。即ち、動作中のOSの動作を中断し、別なO

Sの動作を再開させる。更に、OS間制御ソフトウェアは、動作している各OSの起動、停止を監視し、かつ各OSを単独で起動、停止する機能を持つ。このため、制御装置内のハードウェア、ソフトウェアの部分的な初期化が可能であり、障害部位の自動復旧等の高信頼化が可能となる。

【0007】また、物理メモリには、各OSで占有して使用する空間と、複数のOSから共通してアクセスできる空間とを設定する。これによりカーネルおよびプログラムの個別のメモリ空間については異なるOS間でのデータ破壊等の干渉を防止する一方で、必要なデータを異なるOS上で動作するプログラム間で共有する。更に、あるOS上で動作するプログラムが、他のOSあるいはその上で動くソフトウェアでのイベント発生通知を待機し、これをOS間制御ソフトウェアが通知する機能を持つ。これにより異なるOS上で動作するプログラム間の通信機能が実現される。

【0008】

【発明の実施の形態】図1に本発明の実施例の一つである制御装置の構成図を示す。

【0009】制御装置1のハードウェアは中央演算処理ユニット(CPU)11、物理メモリ13、タイマ14、ユーザ入出力装置16、ネットワーク装置18、割込みコントローラ19から構成される。これらは一般的なパーソナルコンピュータ(PC)と同一の構成である。制御装置1はネットワーク装置18を持ち、これをネットワーク3に接続して、他の制御装置との連携および上位管理計算機に対する制御結果報告に使用する。また制御装置1は入出力装置12を持ち、これを制御対象装置2と接続して信号を入出力し、制御対象装置2からの情報を取得したり、制御対象装置2に動作を指示したりする。制御装置1の割込みコントローラ19は、制御装置1が持つ各構成要素からの割込み信号を一旦受付けてCPU11に伝達したり、あるいは割込み信号をマスクしたりする。

【0010】制御装置1のソフトウェアは、ハードウェア資源管理および上位で動作するプログラムの実行管理を行うオペレーティングシステムA(OS-A)21、オペレーティングシステムB(OS-B)22、これらのOS-AとOS-Bの動作切替え等を行うOS間制御ソフトウェア23、OS-A上で動作する制御監視プログラム24および開発環境プログラム25、OS-B上で動作する制御プログラム26から構成される。OS-A、OS-B、OS間制御ソフトウェア23は、全て単一のCPU11上で特権モードで動作し、CPU11自体の制御等を行う特権命令まで含めた全命令を実行できる。一方、各OS上で動作するプログラムはCPUの非特権モードで動作し、特権命令は実行できない。なお、OS-AおよびOS-Bには、OSのカーネルの他に、特権モードで動作するデバイスドライバが存在するが、

以下ではこれらをカーネルと一括して、「OS」として説明する。

【0011】入出力装置12はOS-Bで独占的に管理・使用する。一方、OS-A上で動作するプログラムが使用するユーザ入出力装置16、ネットワーク接続装置18は、OS-Aで独占的に管理・使用する。ただし、非常停止ボタン・警報出力などの緊急性を要するユーザ入出力装置または、他制御装置との連携等で実時間性を必要とするネットワーク接続装置をOS-Bで管理・使用し、OS-B上のプログラムからアクセスする構成もありうる。物理メモリ13は原則としてOS-A用131と、OS-B用132とに領域を分けて使用する。OS間制御ソフトウェア23が直接アクセスするハードウェアはタイマ14と割込みコントローラ19に限定される。

【0012】最初に、ソフトウェアの動作の概要と、ハードウェアとの関係を説明する。本実施例では、OS-AとしてPCで普及している汎用OSを使用する。OS-Aでは、制御監視プログラム24および開発環境プログラム25に適した高機能なユーザインタフェースを提供する。OS-A及びOS-A上で動作するプログラムは、OS-Bが存在しないとときと同様に動作する。その動作中にOS間制御ソフトウェア23が割込み、OS-Aを中断してOS-Bに切替えることがあるが、OS-Bの動作終了後、OS間制御ソフトウェア23がOS-Aの中断状態を復旧し動作を再開させるため、OS-A側ではOS-Bの存在を意識する必要はない。

【0013】一方、OS-Bは実時間性等に優れ、制御プログラム用に適したOSであるものとする。OS-Bは、OS-B上でのプログラムが処理する内容が無くなった状態(以下、これを「アイドル状態」と呼ぶ)になると、実行権を放棄しOS-Aに実行権を譲る。

【0014】OS間制御ソフトウェア23は、OS-AとOS-Bの実行権の切替えと、両OS上のプログラム間の通信機能を提供する。OS-AからOS-Bへの実行権の切替えは、i) OS-Bの管理するハードウェアからの割込みがあった場合、ii) 予め指定した時間が経過した場合、及び、iii) OS-AからOS-Bに対しての通信があった場合、に行う。一方、OS-BからOS-Aへの実行権の切替えは、前述のようにOS-Bがアイドル状態になった場合にのみ行う。この切替え動作によりOS-BはOS-Aよりも常に優先して実行権が与えられる。OS間制御ソフトウェア23は、切替え動作時およびOS間の通信要求時にのみに動作し、それ以外は、各OSが単独で動作する。CPU11に対する特権命令も各OSが発行し、OS間制御ソフトウェア23がエミュレーションすることはない。なお、OS間制御ソフトウェア23は、いずれのOSの実行時にも、特権モード命令を実行できる各OSの一部として動作することが必要で、特に汎用OSであるOS-Aの動作中は、O

S-Aのデバイスドライバとして動作するように組み込む。

【0015】以上述べたように、OS-AとOS-Bには、各々独占管理するハードウェアが割当てられ、OSの実行権は交互に切替えられて、切替え後は各OSは独立して動作する。OS間制御ソフトウェア23は、両OSの動作の切替えを行い、また両OS間の通信機能を提供する。

【0016】次に、OS間の独立性を実現する方法について説明する。第1に、各OSが独占的に制御し管理するハードウェアが、他方のOS側からの干渉を受けないようにする方法を述べる。各ハードウェアとソフトウェアとのやりとりは、各ハードウェア毎に決められたI/Oアドレスに対する入出力処理と、各ハードウェア毎に決められた割り込み番号の割り込みに対するソフトウェアの応答処理からなる。従って、一方のOSで独占的に制御し管理するハードウェアに割当てられたI/Oアドレスへの入出力及び同ハードウェアに割当てられた割り込み番号に対する応答を、他方のOS側で実行しないように保証することで、ハードウェア制御に関するOS間の独立性を達成する。

【0017】このために、各ハードウェア毎に、どちらのOSで管理・使用するかを、そのハードウェアに割当てられたI/Oアドレス及び割り込み番号と合わせて、OS間制御ソフトウェア23の内部にある割り込みテーブルに記憶する。OS間制御ソフトウェア23は、OS-Aのデバイスドライバとして制御装置1の起動時に初期化処理を実行する。その初期化処理の中でOS-B用ハードウェアのI/Oアドレス及び割り込み番号を予約し、先の割り込みテーブルに登録する。これにより、OS-AのカーネルおよびOS-Aの他のデバイスドライバからは、該当I/Oアドレス及び割り込み番号が使用中と認識されることになり、OS-Aからこれらハードウェアへの処理は行われなくなる。一方、OS-Bでは、OS-A側で使用するハードウェアのI/Oアドレスと割り込み番号を、OS間制御ソフトウェア23から直接取得して、同様に先の割り込みテーブルに登録することにより、これらハードウェアへの処理を禁止する。以上の処理により、ハードウェア制御に関するOS間の独立性を達成できる。

【0018】なおハードウェアに対する入出力処理自体はOS-A、OS-Bが各々直接実行し、OS間制御ソフトウェア23がエミュレートすることはない。これにより、入出力処理のエミュレーションにより余分なオーバーヘッドが発生するのを防ぐと同時に、ハードウェアへの直接アクセスを前提として作られたOSのカーネルおよびデバイスドライバに対する変更を行わずに済む。

【0019】第2に、各OSが使用する物理メモリ空間の独立性を実現する方法について述べる。物理メモリ13については、OS-A、OS-Bで使用する領域を予

め分けて使用することで、独立性を達成する。また、両OS間のデータ共有等のため、OS-AおよびOS-Bで共用するメモリ領域も設定する。

【0020】OS-A、OS-Bからの物理メモリ空間の使用領域を分けるために、制御装置1起動時に、まずOS-Aを起動し、OS-Aの起動オプション指定によって一定のアドレスより下位の物理メモリのみを使用するように設定する。OS-Aの起動後、OS-Aが使用しない前記一定アドレスより上位の物理メモリ領域にOS-Bをロードして起動する。OS-Bのカーネルは、OS間制御ソフトウェア23上のテーブルに記録された前記一定アドレスを取得し、以後のメモリ管理でOS-Aが使用しない領域のみを使用する。

【0021】一方、OS間で共用するメモリ領域には二つの場合がある。一つには前記OS-A用のメモリ領域をOS-B用のアドレス空間にマッピングすることにより実現する。この場合、OS間制御ソフトウェア23が、OS-Bカーネルのメモリ管理機構に依頼して、OS-B用の物理メモリ論理アドレス変換テーブル（以下、これを「ページテーブル」と呼ぶ）に、共用する物理メモリのアドレスを追加し、物理メモリと論理アドレスの対応付け（以下、これを「メモリマッピング」または単に「マッピング」と呼ぶ）を行うことにより実現する。もう一つには、前記OS-B用のメモリ領域をOS-A用のアドレス空間にマッピングすることにより実現する。この場合は、OS間制御ソフトウェア23が、OS-Aのデバイスドライバとして、共用する物理メモリのアドレスを予約する。これはOS-Aが、メモリマッピング型の物理デバイスのデバイスドライバを実現するために備えている機能を使用する。これにより、OS-AのカーネルがOS-A用のページテーブルに、共用する物理メモリのマッピングを行い、該当領域の共用が実現できる。

【0022】第3に、各OSの論理アドレス空間の独立性を実現する方法について述べる。OS並びにOSの上で動作するプログラムは、論理アドレスによってメモリにアクセスし、その論理アドレスから物理アドレスへの変換は、CPUがページテーブルを参照して自動的に行う。OS-A、OS-Bの各々に存在するメモリ管理機能がこのページテーブルを操作し、マッピングを行う。この際、両OSが使用するページテーブルを分離し、OSの実行権に応じて切替えて使用することにより、各OSごとにマッピング操作を独立に行うことができる。

【0023】ページテーブルは、物理メモリ上に存在するテーブルだが、CPUにはそのテーブルの先頭物理メモリ位置を指示するレジスタが存在する。CPUはこのレジスタからページテーブルの位置を求め、その内容に従ってアドレス変換を自動実行する。従って、OS-A用とOS-B用のページテーブル領域を物理メモリ上別個に用意し、OS間制御ソフトウェア23がOSを切替

える処理の中で、前記ページテーブル位置を示すレジスタの内容を切替えて、切替え先OSのページテーブルを指すように書換える。これにより両OSは各々のページテーブルでマッピングが実行でき、論理アドレス空間の独立性を確保できる。

【0024】ここで説明した物理メモリ空間および論理アドレス空間の使用方法を図2に示す。物理メモリ空間51をいくつかの領域に区切り、これをOS-Aカーネル用52、OS-Aプログラム用54、OS-Bカーネル用56、OS-Bプログラム用57に分けて、OS-A、OS-Bの各々割当て先からアクセスできるようにする。OS-A、OS-Bから認識されるメモリの論理アドレス空間61、62は、各々のOSが持つページテーブルにより、独立した論理アドレス空間となっている。OS-A動作時は、OS-B専用の物理メモリ空間56、57が論理アドレス空間61にマッピングされない。これにより、OS-Aの動作中にOS-B用の領域は一切アクセスされず、データ誤破壊等を防ぐことができる。OS-Bの動作中は逆にOS-A専用の物理メモリ空間52、54を論理アドレス空間62にマッピングしない。OS間制御ソフトウェア23はどちらのOSの動作中も動作しなければならないので、そのメモリ領域53はいずれのOSでもマッピングされる。また、OS-A・OS-B共通領域55は、いずれのOSでもマッピングされ、各OS上のプログラムからアクセスすることが可能で、両OS上プログラム間のデータ交換に使用できる。ここで、OS-A・OS-B共通領域は非特権モードのプログラム用に限定することで、各OSのカーネルおよびデバイスドライバに対して、もう一方のOSおよびプログラムから影響を与えることがなく、両OSの独立性・信頼性を維持できる。なお、各領域は物理メモリ空間51上で、CPU11のアドレス変換機構の管理単位で断片化し分散して配置されていてもよい。

【0025】以上述べた方法により、OS間の独立性が確保される。

【0026】次に、OS間制御ソフトウェア23の動作を詳細に説明する。OS間制御ソフトウェア23は、各OSまたはプログラムから明示的に呼出された場合、およびCPUに割込みが入力された場合に動作が開始される。OS-Aあるいはその上で動作するプログラムからの呼出しは、OS間制御ソフトウェア23がOS-Aのデバイスドライバとして組み込まれているため、該当デバイスドライバに対する動作指示（IOCTL命令など）として実現する。また、OS-Bあるいはその上で動作するプログラムからの呼出しは、OS-BがOS間制御ソフトウェア23の存在を認識しているため、OS-Bのカーネル内の処理から関数呼出しにより呼出す。一方、割込み発生時は、CPU11が物理メモリ13上に存在する割込みテーブルを参照して割込みハンドラを呼出すことから、前記割込みテーブルを変更して全ての

割込みのハンドラをOS間制御ソフトウェア23内のルーチンとする。これにより、全ての割込み発生時にOS間制御ソフトウェア23が動作し、適切な処理を行う。

【0027】起動されたOS間制御ソフトウェア23は、起動した要因に応じて、OSの動作の切替え、両OS間の通信に必要な処理等を行う。以下、詳細な処理ステップを記す。

【0028】図3は、実行が非優先のOSであるOS-Aの動作中に割込みが入力された場合の処理を示す。割込み入力の結果、OS間制御ソフトウェア23の割込み処理ルーチンが呼出され、図示された処理を実行する。まず、割込み時点でのレジスタ内容をスタック上に保存してスタックフレームを作成する(S01)。ついで、割込み番号(ベクタ)により、この割込みがソフトウェア割込みなのか、ハードウェア割込みなのかを判定する(S02)。ソフトウェア割込みの場合は、その時点で動作中のOS-A側の処理で明示的に割込み命令を発行したか、またはプログラムの動作により例外が発生した場合であり、いずれにしてもOS-Aの本来の割込みハンドラにジャンプする。一方、ハードウェア割込みの場合、どのハードウェアから発生した割込みか割込み元を判定する(S03)。OS-A側が管理するハードウェアから発生した割込みである場合には、ソフトウェア割込みと同じく、OS-Aの本来の割込みハンドラにジャンプする。しかし、OS-B側が管理するハードウェアから発生した割込みであった場合には、まず動作環境をOS-Bに切替え(S04)、OS-Bの本来の割込みハンドラへとジャンプする。又、割込みがタイマ割込みの場合、割込み発生時刻を判定し(S05)、OS-AまたはOS-Bのタイマがタイムアップすべき時刻だった場合、各OSの本来のタイマハンドラにジャンプする。ただしどちらもタイムアップの時刻だった場合は、OS-Aのタイムアップは保留し、優先されるOS-Bのハンドラ呼出しのみを行う。

【0029】図4は、実行が優先されるOSであるOS-Bの動作中に割込みが入力された場合の処理を示す。この場合もOS-Aの動作中に割込みが入力された場合の処理と同様に、割込み入力の結果、OS間制御ソフトウェア23の割込み処理ルーチンが呼出され、図示された処理を実行する。まず、割込み時点でのレジスタ内容をスタック上に保存する(S11)。ついで、割込み番号(ベクタ)により、割込みがソフトウェア割込みなのか、ハードウェア割込みなのかを判定する(S12)。ソフトウェア割込みの場合、動作中のOS-Bの割込みハンドラにジャンプする。ハードウェア割込みの場合、どのハードウェアから発生した割込みか判定する(S13)。OS間制御ソフトウェア23は、OS-Bの動作に切替える際に割込みコントローラ19を操作して、OS-Aが管理するハードウェアからの割込みをマスクする。このため、OS-Bの動作中はOS-Aが管理するハード

ウェアからの割込みは発生しない。OS-B側が管理するハードウェアから割込みが発生した場合は、ソフトウェア割込みと同じく、OS-Bの本来の割込みハンドラにジャンプする。しかし、割込みがタイマ割込みの場合、割込み発生時刻を判定し(S14)、OS-Bのタイマがタイムアップすべき時刻だった場合、OS-Bの本来のタイマハンドラにジャンプする。OS-Aのタイマがタイムアップだった場合は、そのタイムアップ発生を記録し(S15)、この時点ではOS-Bの動作に復帰する。なお、割込みコントローラ19の操作が不可能なハードウェア構成のため、割込元判定(S13)の際にOS-A管理機器と判定された時にはその割込み発生を記録し、後でOS-Aに動作を切替える際にOS-Aの本来の割込みハンドラを呼出せばよい。

【0030】以上の説明は、OS-Bを完全に優先させる場合の動作である。しかしこの処理では、OS-BからOS-Aへの動作切替えが行われるのは、OS-Bがアイドル状態となって、OS-BからOS間制御ソフトウェア23に対してアイドル状態を通知する呼出しが行われた時のみとなってしまう。そこで、OS-B側で無限ループに陥った場合のデッドロック回避などを考慮し、一定の割合でOS-Aの動作を行わせる。そのため、タイマ割込みをOS-A、OS-Bへのタイムアップ時刻以外にも発生させて、OS間制御ソフトウェア23の割込み処理中でOSの切替えを行う。OS間制御ソフトウェア23では、OS切替え処理を行う時点で、予め各OSの実行時間を計算しておく。そして、OS-B動作中にタイマ割込みが発生した場合には、計算しておいたOS-Bの実行時間を調べ、OS-Aに切替えるべき時間が経過したか判断し(S16)、OS-Aへの切替え時間が経過していたならばOS-Aの動作環境に切替えた後(S17)、OS-Aの中断点にジャンプする。又OS-Aへの切替え時間が経過していなければ、そのままOS-Bの割込み発生時の中断点に復帰する。逆にOS-A動作中にタイマ割込みが発生した場合、図3に示す処理において、OS-Bがアイドル状態か否かを判定し(S06)、OS-Bがアイドル状態でない場合には更に、OS-Bに動作を戻すべき時間が経過したか判断する(S07)。OS-Bへの動作切替え時間が経過していれば、OS-Bの動作環境に切替えて(S08)、OS-Bの中断点にジャンプする。しかし、OS-Bがアイドル状態であるか又はOS-Bへの動作切替え時間が経過していなければ、そのままOS-Aの割込み発生時の中断点に復帰する。

【0031】図5に、OS間制御ソフトウェア23における、各OSへの切替え処理の詳細を示す。OSの切替え処理は、前記の割込み処理の過程で切替えが必要と判断された場合、又はOS-Bがアイドル状態となった場合、あるいは後述するイベント発生通知でOS-B側の待機プログラムの待機解除が必要になった場合に行われ

る。まず割込み発生時点またはOS間制御ソフトウェア23を呼出した時点(以下、これを「中断点」と呼ぶ)のコンテキストを保存する(S31)。即ち、中断点におけるCPU11のレジスタ内容を保存したスタックフレーム位置および、命令カウンタ、スタックポインタ等の値を記録する。次に割込みコントローラ19の設定を変更し、OS-B側動作中にOS-A側割込みが発生しないようにマスクする、又は逆にOS-A側割込みに対するマスクを解除する操作を行う(S32)。

【0032】次いでページテーブルの先頭メモリ位置を示すレジスタを変更してメモリ空間を切替える(S33)。この操作は既に述べた通りである。次にOS間のイベント発生通知の処理を行う(S34)が、この詳細は後述する。OS-BからOS-Aに切替える時には、図4に示す処理のS15で発生が記録されている割込みに対応する割込みルーチンを呼出して、保留している割込み処理を行わせる(S35)。この後、記録している割込みを全て処理したならば(S36)、中断時にS31で保存されたコンテキストを回復し(S38)、切替えるOSの中断点に復帰する。ただし、OS-Bがアイドル状態でOS-Aの切替え要求を行う際に、レジスタ内容を放棄してもいいように必要なデータをすべてメモリ上に保存する構成とし、OS-Bの割込みハンドラにジャンプする場合は、コンテキストは回復しない(S37)。

【0033】なお割込みコントローラ19は、割込み番号毎に割込みのマスクができるが、OS-Aが単独で動作する場合、OS-Aの割込み処理中にはより優先度の低い割込みがマスクされる。このマスクされた割込みにOS-B用の割込みが含まれる場合、マスクが解除されるまで該当割込みの処理に遅延が生じる。これを防止するために、OS-Aカーネルの割込みコントローラ処理のうち、OS-B用の割込み番号に対しては、操作を行わないように修正する必要がある。一方、OS-Aのタイマ割込みは、一定周期でタイマ14から割込み信号を受け、その処理を行うだけであり、初期化時以外、OS-Aの動作時はタイマ14を操作しない。従って、タイマ14からの割込み信号は一旦OS間制御ソフトウェア23で受け、図3に示したS03以降の処理により、OS-Aの割込みハンドラ呼出しをするだけでよい。

【0034】また、OS間制御ソフトウェア23は、2つのOS間の通信機能のために、いずれかのOSから明示的に呼出される場合がある。2つのOS間通信のための基本機能である、OS間でのイベント発生通知機能を図6に示す。

【0035】OS間制御ソフトウェア23の管理するメモリ上に、OS-Aイベント処理テーブル71が存在する。処理可能なイベントにはi1, i2, ... iNのイベント番号が付けられ、対応したN個のエントリが存在する。初期状態でイベント処理テーブルは空である。OS-A側のプログラム(プログラム-A)がイベント番号

(i2)を指定して、イベントの発生を通知して待機要求をOS間制御ソフトウェア23に出す(S41)と、OS間制御ソフトウェア23はテーブル71の該当イベントのエントリi2に、このプログラム(プログラム-A)が待機中であることを記録する。待機要求を出したプログラムはOS-Aのプログラム中断機能により実行を停止する。次にOS-B側のプログラム(プログラム-D)からイベント番号(iN)を指定してイベント発生通知(S42)があると、OS間制御ソフトウェア23はテーブル71の該当エントリiNを見て、OS-A側で待機しているプログラムの有無を判定する。待機しているプログラムが有る場合には、OS-Aにそのプログラム(プログラム-B)の待機解除要求を行う(S43)。これにより、待機していたOS-A側のプログラムは実行を再開し、待機していたイベントが発生したことを知る。また、イベントの発生通知は、OS-Aで動作するプログラム(プログラム-C)からでも同様に可能(S44)である。このようにすることで、同一OS上で発生するイベントと異なるOS上で発生するイベントを同時に待機することが可能となる。同じ様に、OS-B用のイベント処理テーブル72が用意されており、前記説明と対称的にOS-B側のプログラムでイベント待機をすることが可能である。

【0036】前記イベント通知で使用するプログラムの中断と再開機能は、OS-A側では、デバイスドライバとして動作しているOS間制御ソフトウェア23が、デバイスに対する入出力を開始および終了したことを報告する機能を使い実現できる。OS-B側では、OS間制御ソフトウェア23が、直接プログラムの中断・再開を行うOS-Bのルーチンを呼出すことで実現する。前記イベント通知の説明では待機する単位はプログラムと記述したが、マルチスレッド環境を提供するOSでは、動作単位であるスレッドの中断と再開が行われる。またイベント処理テーブルにはプログラムの番号を格納すると記述したが、代わりに待機しているものを判別するための、OS内構造体やそのポインタを格納してもよい。さらに、OS間制御ソフトウェア23の管理するメモリ上に別途バッファを設けて、イベント通知側から発生通知と同時にデータを受け取りバッファに記録し、待機解除時にデータを渡すように構成すれば、到着待機機能を持つメッセージ通信機能が実現できる。

【0037】さらに、OS間制御ソフトウェア23は、OS-AあるいはOS-BのいずれかのOSを動作させたまま、もう一方のOSを停止、再起動する機能を提供する。

【0038】第1に、OS-Aを動作させたまま、OS-Bを停止、再起動する処理について説明する。OS-Bがシャットダウンする場合、または例外が発生し強制停止する場合、OS間制御ソフトウェア23のルーチンを呼出し、これを通知する。OS間制御ソフトウェア2

3では、以後OS-Bに動作切替えを行わないようにするので、OS-Aのみ継続動作する。また、OS-BはOS間制御ソフトウェア23から所定の初期化ルーチンが呼出されることにより起動するので、再起動は通常起動と全く同様にできる。

【0039】第2に、OS-Bを動作させたまま、OS-Aを停止、再起動する処理について説明する。OS-Aのシャットダウン時は、OS-A上のプログラムでシャットダウンの実行を検出してOS間制御ソフトウェア23に通知する。又OS-Aの例外発生時は、OS間制御ソフトウェア23の割込みハンドラで判定する。これにより、OS-Aのシャットダウン時または例外発生時には、OS間制御ソフトウェア23ではOS-Aへの動作切替えを行わないようにするので、OS-Bのみが継続動作することになる。ところが、停止したOS-Aを通常の起動の場合と同様に再起動すると、バスなど、OS-Bも使用している共通ハードウェアの初期化も行ってしまう、OS-Bの継続動作を妨げる場合がおこる。そこで、OS-Aの停止後再起動は図7に示した手順で行う。電源投入時、OS-Aは共通ハードウェアの初期化(S51)を行う。その後、デバイスドライバとして動作しているOS間制御ソフトウェア23は、OS-Aが管理するハードウェアの初期化処理のタイミングを与えられるが、ここでコンテキストの保存(S52)を行う。このコンテキストの保存では、レジスタの内容の他、OS-Aが管理する全メモリ領域の内容を、OS間制御ソフトウェア23の管理するメモリ領域にコピーする。その後、OS-Aが管理するハードウェアの初期化(S53)が、OS-A側の各デバイスドライバによって行われる。この初期化処理後、OS-Aは通常運用の状態になるが、OS-Aの通常運用中、シャットダウンまたは例外発生によってOS-Aが動作停止した場合、OS間制御ソフトウェア23は前述の通りOS-Aのみの停止を行う(S54)。その後、OS間制御ソフトウェア23が自動で、あるいはOS-B側のプログラムからの要求で、OS-A側の保存しておいたコンテキストの復旧を行う(S55)。すなわちOS-Aが管理するハードウェアの初期化処理の際(S52)に保存したコンテキストにより、初期化時点のメモリ内容を再現し、又レジスタ内容を復旧し、この時点でOS-Aへの動作切替えを再開する。これによりOS-Aは共通ハードウェアの設定が終了した直後から実行を再開し、OS-Aが管理するハードウェアの初期化のみを行った後、再び通常運用に戻るようになるため、OS-Bも使用している共通ハードウェアの初期化を再度行うことが無い。

【0040】本実施例では、汎用OSであるOS-Aに関して、割込みコントローラ19に対する処理を修正するだけで、それ以外に変更は加えていない。これは、汎用OSのOS-Aに対して、OS-BおよびOS間制御ソフトウェア23の追加を非常に容易にしている。

【0041】

【発明の効果】以上のように、本発明によれば、制御装置内の単一のCPU上で複数のOSを動作させる際に、OSおよび他プログラムの異常の影響を局所化し、またOS単位での部分的な再起動により制御装置全体の動作を停止せずに正常動作に復旧させることが可能となり、信頼性が向上する。また、通常のOSおよびプログラム動作空間から独立した空間で、OSの動作を監視することが可能となり、信頼性が向上する。

【図面の簡単な説明】

【図1】本発明を用いた実施例の構成図である。

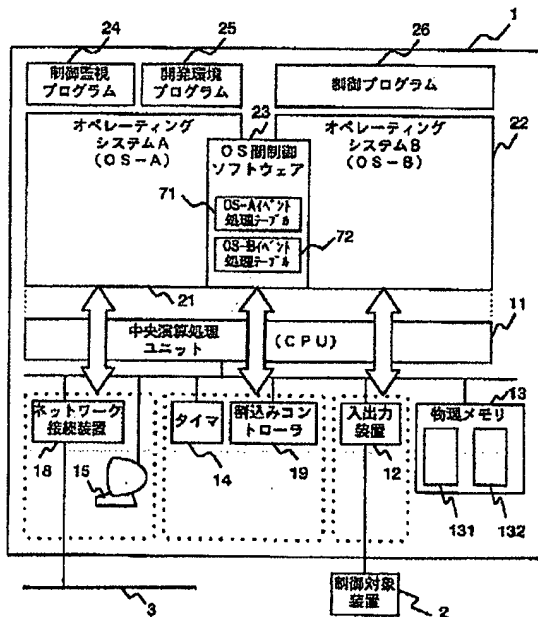
【図2】制御装置内でのメモリの使用状況を説明した図である。

【図3】OS-A動作中の割込み処理手順を説明したフローチャートである。

【図4】OS-B動作中の割込み処理手順を説明したフ*

【図1】

図 1



* ローチャートである。

【図5】OS切替え処理の手順を説明したフローチャートである。

【図6】イベント発生通知機能を説明する図である。

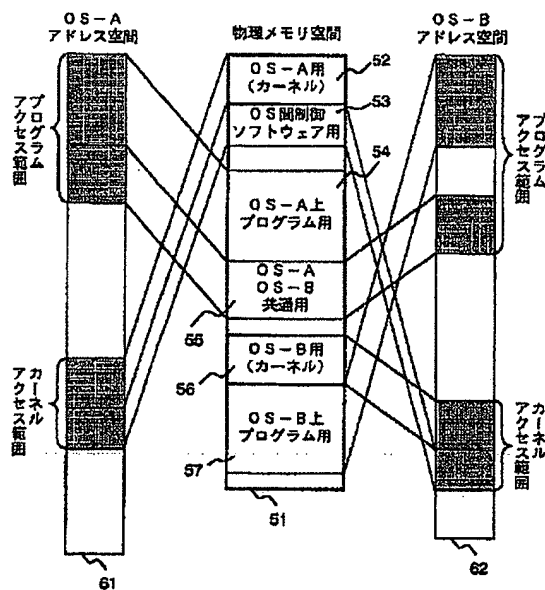
【図7】OS-Aのみを再起動するための処理手順を説明したフローチャートである。

【符号の説明】

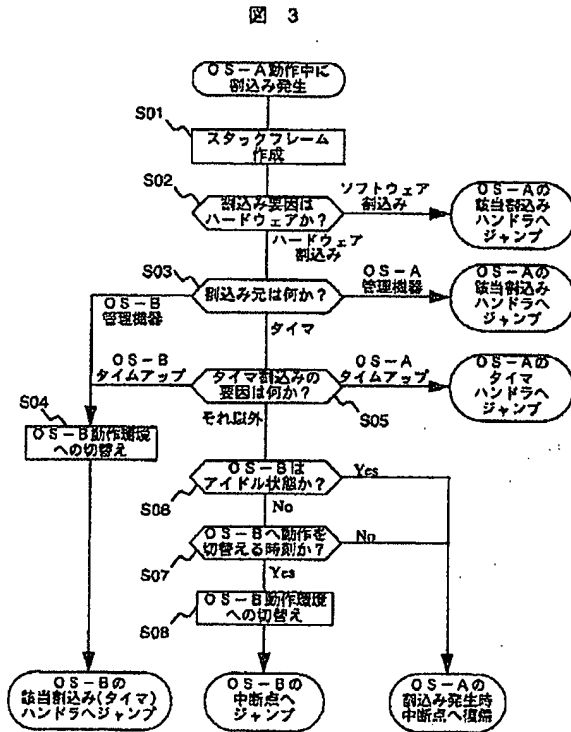
1…制御装置、2…制御対象装置、3…ネットワーク、11…中央演算処理ユニット(CPU)、12…入出力装置、13…メモリ、14…タイマ、19…割込みコントローラ、21…オペレーティングシステムA(OS-A)、22…オペレーティングシステムB(OS-B)、23…OS間制御ソフトウェア、24…制御監視プログラム、25…開発環境プログラム、26…制御プログラム。

【図2】

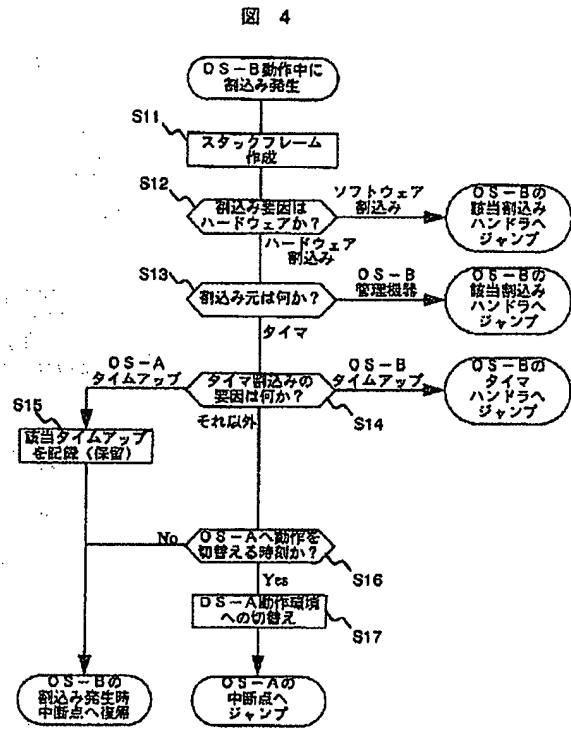
図 2



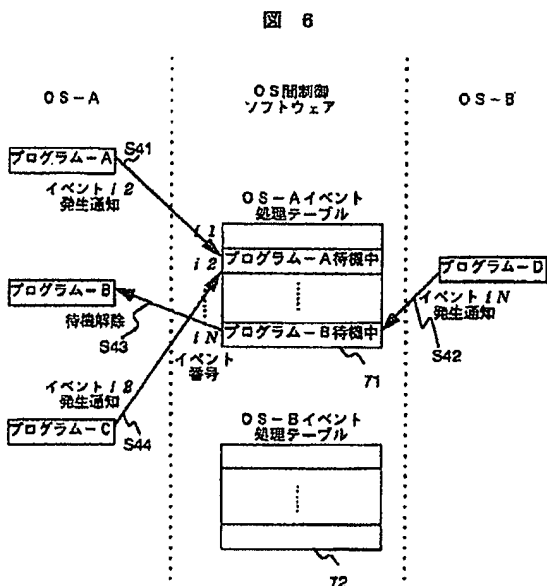
【図3】



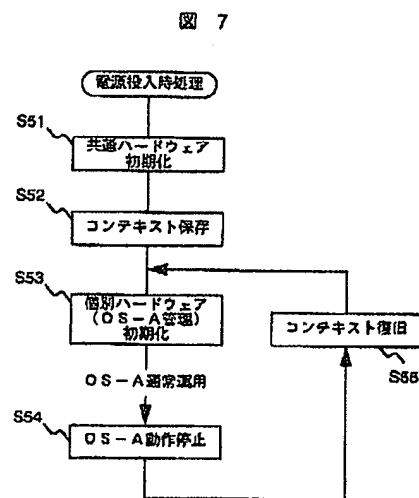
【図4】



【図6】

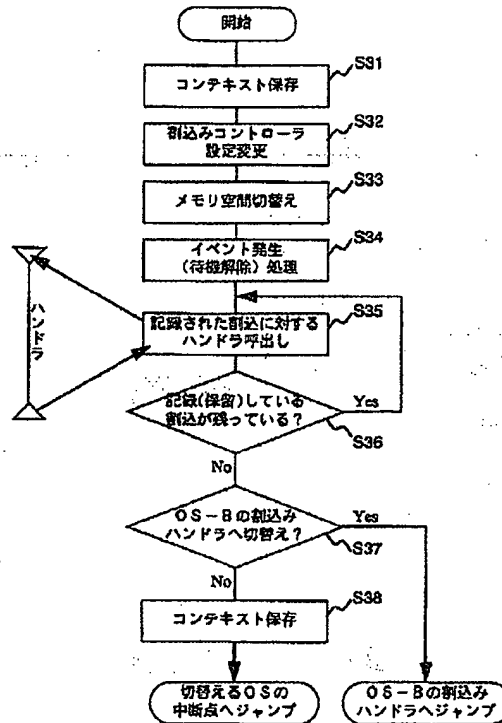


【図7】



【図5】

図 5



フロントページの続き

(72)発明者 金子 茂則

茨城県日立市大みか町五丁目2番1号 株
式会社日立製作所大みか工場内

(72)発明者 吉沢 亮吉

茨城県日立市大みか町五丁目2番1号 株
式会社日立製作所大みか工場内

(72)発明者 加藤 直

茨城県日立市大みか町五丁目2番1号 株
式会社日立製作所大みか工場内

(72)発明者 山内 学

茨城県日立市大みか町五丁目2番1号 株
式会社日立製作所大みか工場内

(72)発明者 新井 利明

神奈川県川崎市麻生区王禅寺1099番地 株
式会社日立製作所システム開発研究所内

(72)発明者 関口 知紀

神奈川県川崎市麻生区王禅寺1099番地 株
式会社日立製作所システム開発研究所内

Fターム(参考) 5B098 BA06 BB03 BB06 EE02 GA02

GA07 GB01 GC03 GC05 GC16

GD03 GD20